



EXPLORING OUR BUGS

BEN COTTON

Fedora Program Manager

✉ bcotton@fedoraproject.org

🐦 [@funnelfiasco](https://twitter.com/funnelfiasco)

[m] [@funnelfiasco:matrix.org](https://matrix.org/#/room/#funnelfiasco:matrix.org)

HOUSEKEEPING

License: CC BY-SA 4.0

Nice things: @FunnelFiasco

Not-nice things: /dev/null

@FunnelFiasco

CC BY-SA 4.0



WHAT IS THIS TALK?

CC BY-SA 4.0

A LOOK AT FEDORA BUGS

- From Fedora Linux 19 through Fedora Linux 32
 - Does not include Rawhide bugs
- Based on curiosity, not convincing
- Asks more questions than it answers

@FunnelFiasco

CC BY-SA 4.0



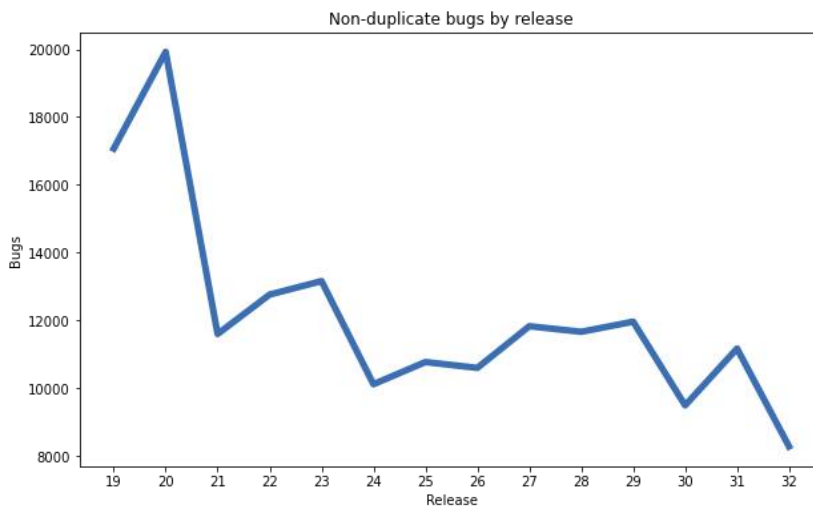
I started with F19 because that was the first release with an EOL closure type. It represents an obvious “modern era” of our bugs. F32 is the last EOL release at the time of this presentation, so we stop there.

I don't include Rawhide for two reasons. First, this talk is more focused on the user view of bugs and Rawhide is (generally) not user-facing. Also, incorporating Rawhide bugs would require adding date-based searches and then shoehorning them into the data set. That's more work than I wanted to put into it.

THE BASICS

CC BY-SA 4.0

BUG REPORTS PER RELEASE



@FunnelFiasco

CC BY-SA 4.0

The obvious place to start is to just look at the number of (non-duplicate) bug reports per release. We can see a general downward trend in time, which may not be good news. Karl Fogel says “an accessible bug database is one of the strongest signs that a project should be taken seriously —and the higher the number of bugs in the database, the better the project looks”.

BUGS PER COMPONENT: TOP 10

Component	Bugs
kernel	9028
selinux-policy	6477
gnome-shell	3645
anaconda	3079
dnf	1925
systemd	1488
firefox	1468
nautilus	1450
xorg-x11-server	1298
PackageKit	1140

@FunnelFiasco

CC BY-SA 4.0

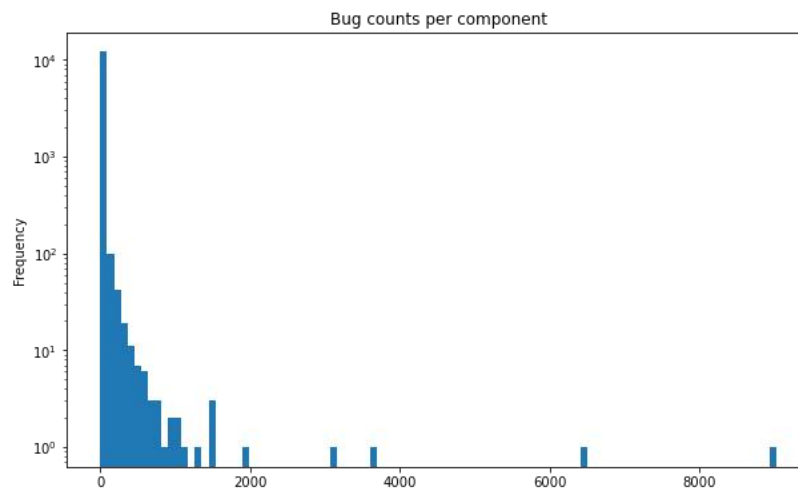


BUGS PER COMPONENT: BOTTOM 10

- 12,082 (98.44%) components with < 100 bugs
- 10,549 (85.95%) components with < 10 bugs



BUGS PER COMPONENT



@FunnelFiasco

CC BY-SA 4.0

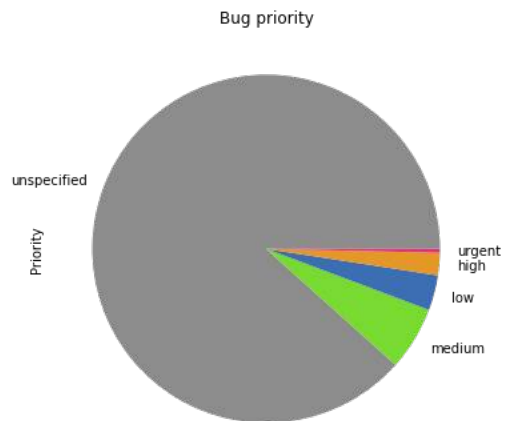
As we might expect, most components have very few bug reports.

PRIORITY AND SEVERITY

CC BY-SA 4.0

Bugzilla says priority is the developer's rating of when it will be worked on and severity is the user's rating of impact. I am not convinced these are used in the way the documentation thinks they should be used.

BUG PRIORITY

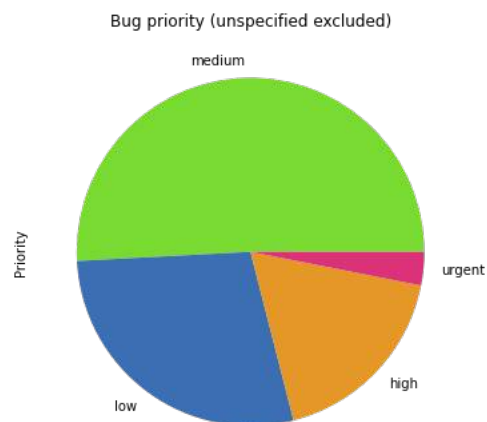


@FunnelFiasco

CC BY-SA 4.0

I expected this to be all “urgent”. But it turns out since Bugzilla does not provide a default value, most people leave it unspecified.

BUG PRIORITY (WHEN SPECIFIED)

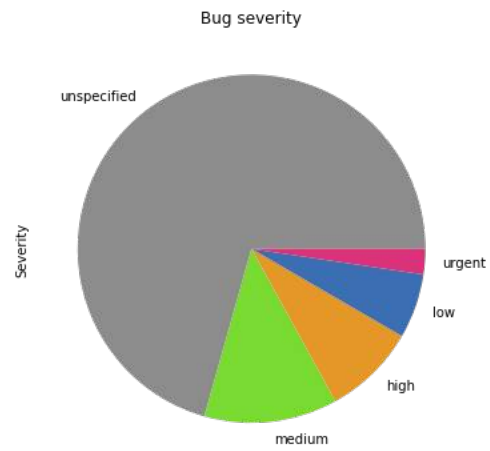


@FunnelFiasco

CC BY-SA 4.0

Removing the unspecified, we see a distribution that makes sense. Urgent bugs represent a small fraction. High is a larger segment, and medium is the highest. The reason I expect medium to be larger than low is that low-priority bugs probably go under-reported.

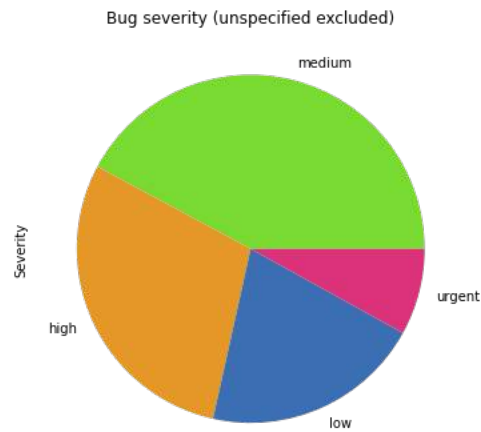
BUG SEVERITY



@FunnelFiasco

CC BY-SA 4.0

BUG SEVERITY (WHEN SPECIFIED)



@FunnelFiasco

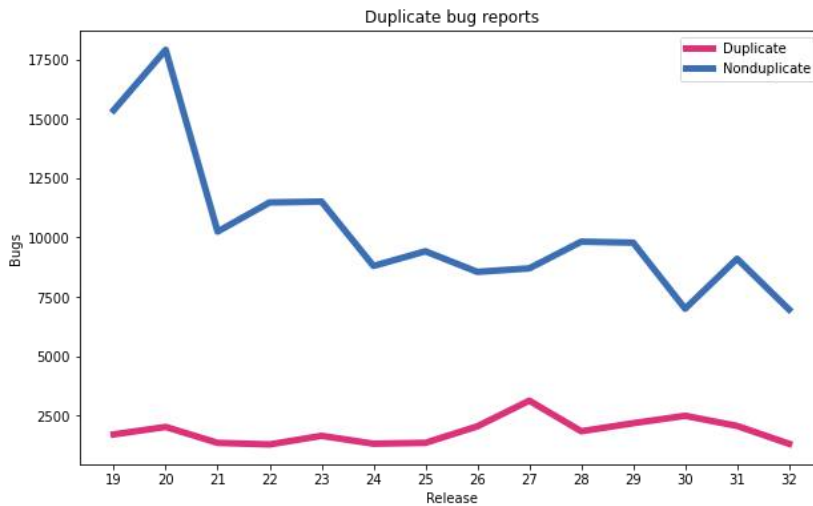
CC BY-SA 4.0

Removing the unspecified, we see a distribution that makes sense. Urgent bugs represent a small fraction. High is a larger segment, and medium is the highest. The reason I expect medium to be larger than low is that low-severity bugs probably go under-reported.

DUPLICATE BUGS

CC BY-SA 4.0

DUPLICATE BUG REPORTS

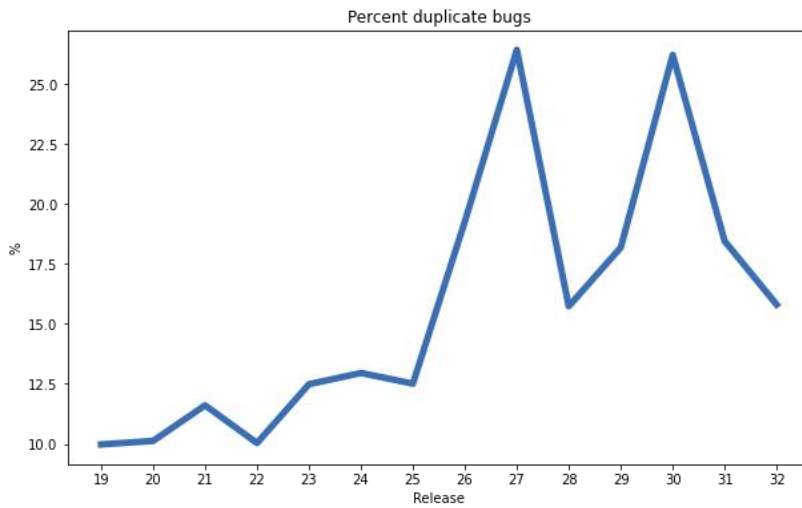


@FunnelFiasco

CC BY-SA 4.0

One thing I wanted to look at was the number of duplicate bug reports. This slide shows the duplicate and non-duplicate bug reports over time. Interestingly, even though the non-duplicate bug counts drop over time, the duplicate bug counts are relatively steady.

% DUPLICATE BUG REPORTS



@FunnelFiasco

CC BY-SA 4.0

...this is reflected clearly when looking at the percentage of duplicates by release.

FEWEST DUPLICATES PER COMPONENT

Component	Duplicates
xen	0.65%
ansible	0.77%
389-ds-base	1.05%
btrfs-progs	1.22%
synergy	1.22%
kubernetes	1.32%
gtk2	1.39%
procps-ng	1.56%
binutils	1.59%
snappd	1.60%

@FunnelFiasco

CC BY-SA 4.0



OOPS! ALL DUPLICATES!

- 63 components have a 100% duplicate count

@FunnelFiasco

CC BY-SA 4.0



I didn't look at all of these components, but I poked at a few. The duplicates were generally repeats of a Rawhide bug or a bug that was also filed against another component.

MOST DUPLICATES PER COMPONENT

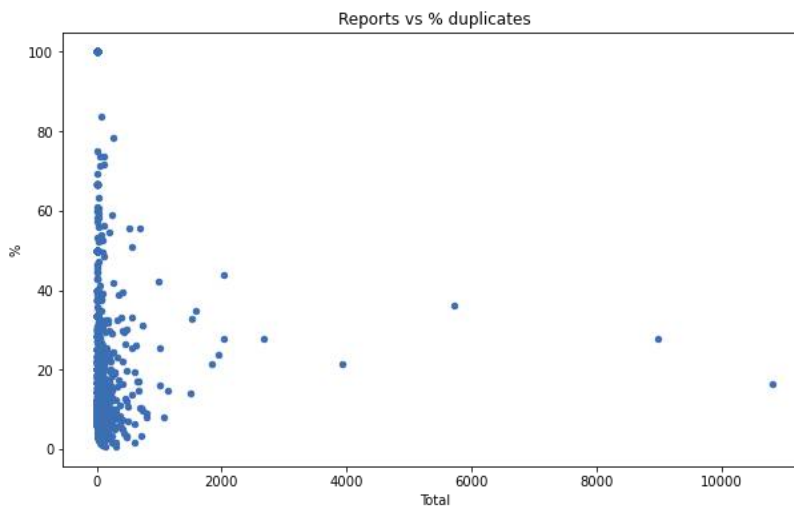
Component	Duplicates
tpm2-arbrmd	83.82%
open-vm-tools	78.46%
pypoppler	75.00%
glib-networking	73.68%
libgnomekbd	73.59%
kf5-kglobalaccel	71.56%
imsettings	71.43%
gq	69.23%
python-PyGithub	66.67%
NearTree	66.67%

@FunnelFiasco

CC BY-SA 4.0



HOW TO GET MORE DUPLICATES?



@FunnelFiasco

CC BY-SA 4.0

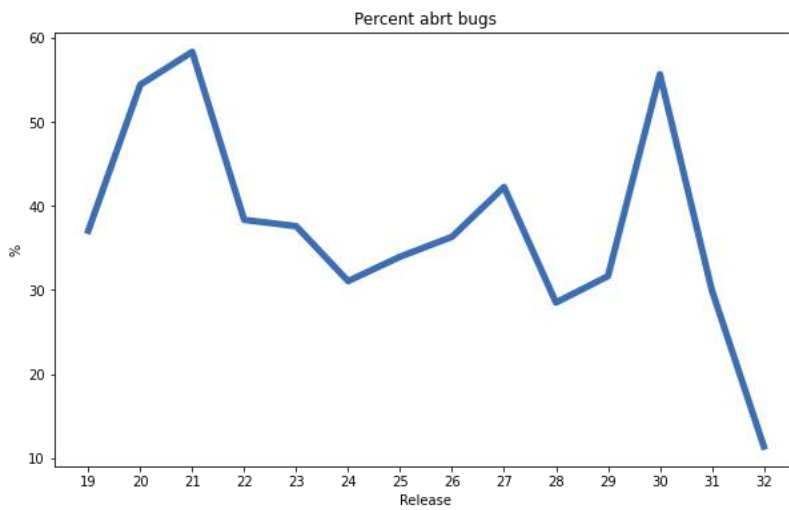
Intuitively, I expected the percentage of duplicates to be relatively steady as the bug count goes up. Above about a thousand bugs or so, that seems to be the case. The dropoff at the end is likely due to duplicates not being marked.

ABRT REPORTS

CC BY-SA 4.0

Matthew Miller wondered how many bugs are from humans versus automated reports. At a first pass, I used abrt (specifically the presence of “[abrt]” in the summary field) as a proxy.

PERCENT OF BUGS FROM ABRT

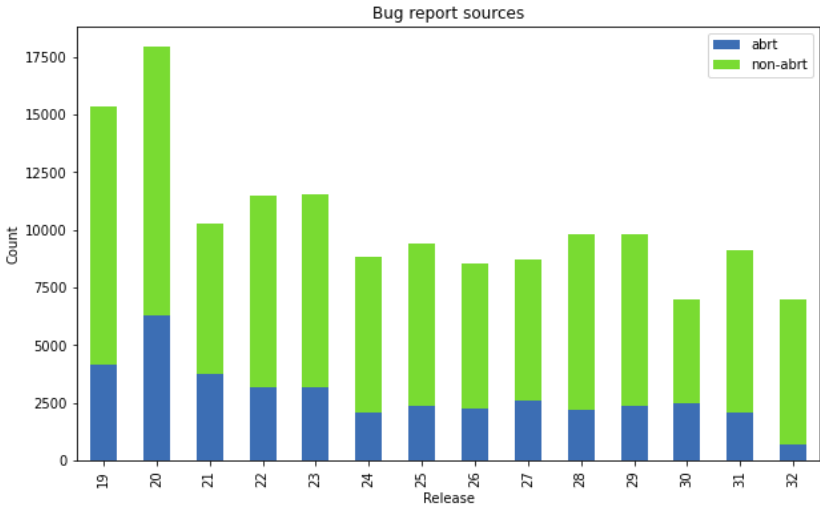


@FunnelFiasco

CC BY-SA 4.0

Fedora Linux 32 had a surprisingly low percentage of abrt bugs.

BUG REPORT SOURCES



@FunnelFiasco

CC BY-SA 4.0



BUG RESOLUTION

CC BY-SA 4.0

CATEGORIZING CLOSURES

- Happy resolutions: CURRENTRELEASE, ERRATA, NEXTRELEASE, RAWHIDE, UPSTREAM
- Sad user resolutions: CANTFIX, DEFERRED, EOL, WONTFIX
- Sad maintainer resolutions: INSUFFICIENT_DATA, NOTABUG, WORKSFORME
- DUPLICATE is excluded here

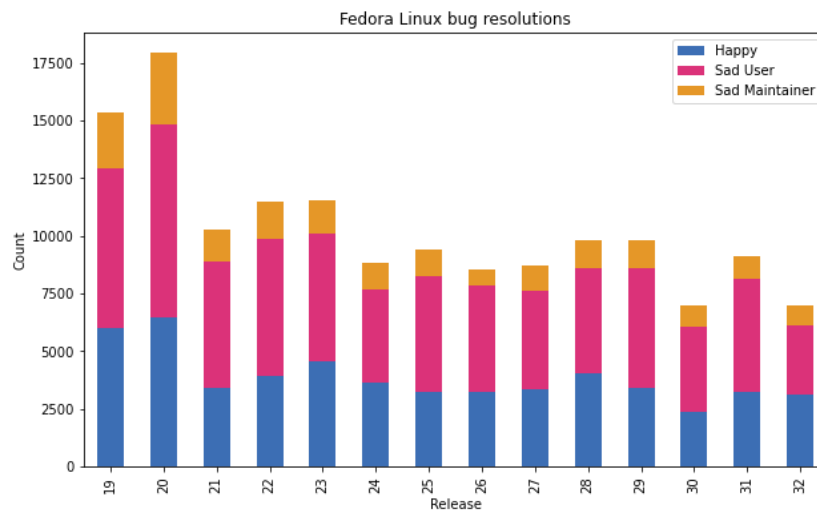
@FunnelFiasco

CC BY-SA 4.0



We can argue about which closure types belong in which category, but this seems reasonable based on how I've seen them used.

CLOSURES BY CATEGORY

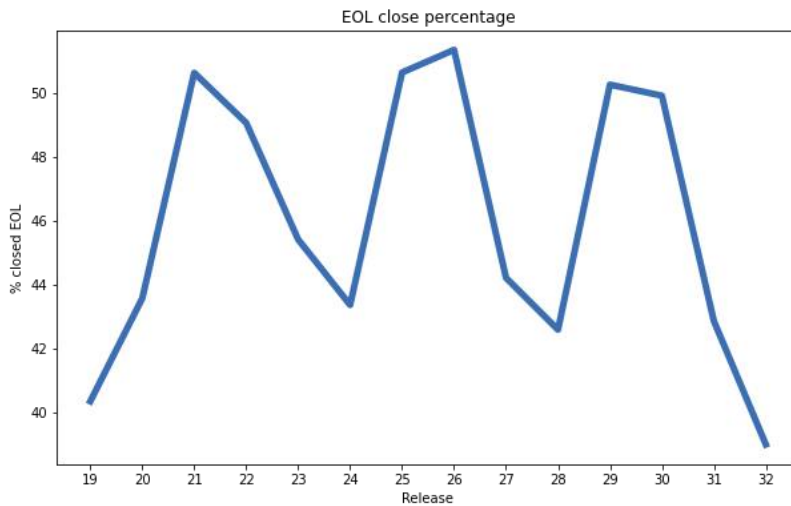


@FunnelFiasco

CC BY-SA 4.0

The happy closures tend to be smaller than I'd like. The sad user closures dominate the sad closures.

EOL CLOSURE PERCENTAGE



@FunnelFiasco

CC BY-SA 4.0

Every six months, we see complaints about how “Fedora never fixes my bugs, I’m done!” So I wanted to look at the closure percentage. Turns out it’s generally 40-50% of bugs that get closed EOL. What’s interesting is the apparent periodicity to the chart. I don’t have a good explanation for that.

LOWEST EOL CLOSURES

Component	EOL closures
curl	0.97%
fedora-obsolete-packages	1.54%
opensc	2.00%
jackson-databind	2.04%
mupdf	2.27%
python-bugzilla	2.94%
gnutls	3.00%
nextcloud-client	3.00%
python2	3.03%
LibRaw	3.13%

@FunnelFiasco

CC BY-SA 4.0



ONLY EOL

- 2,062 components have 100% (non-duplicate) EOL closure

@FunnelFiasco

CC BY-SA 4.0



HIGHEST NOT-100% EOL CLOSURES

Component	EOL closures
xarchiver	98.36%
tracker-miners	98.10%
ekiga	98.00%
kactivitymanagerd	97.83%
telepathy-haze	97.73%
telepathy-idle	96.77%
glade	96.20%
dley-na-renderer	95.96%
matio	95.65%
supertux	95.65%

@FunnelFiasco

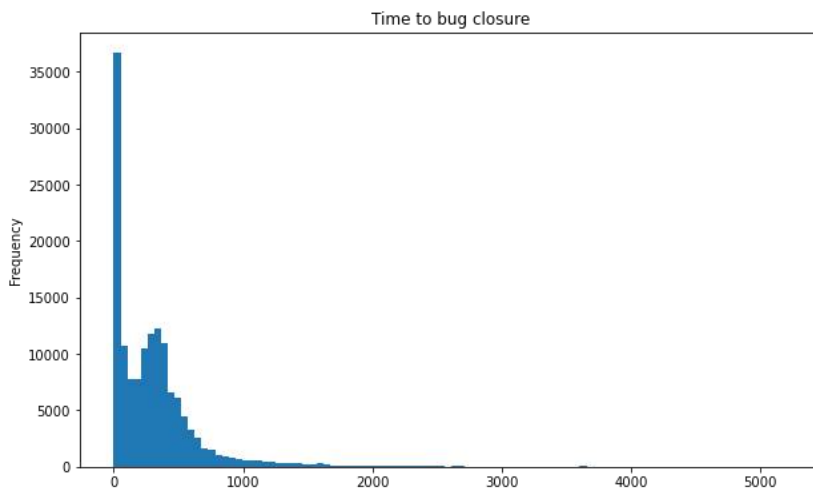
CC BY-SA 4.0



TIME TO RESOLUTION (TTR)

CC BY-SA 4.0

TTR



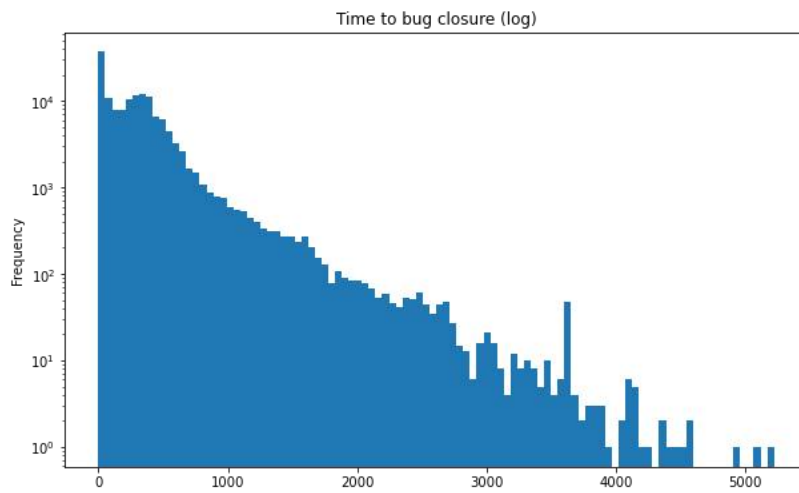
@FunnelFiasco

CC BY-SA 4.0



How quickly are bugs resolved one way or another?

TRR (LOG)

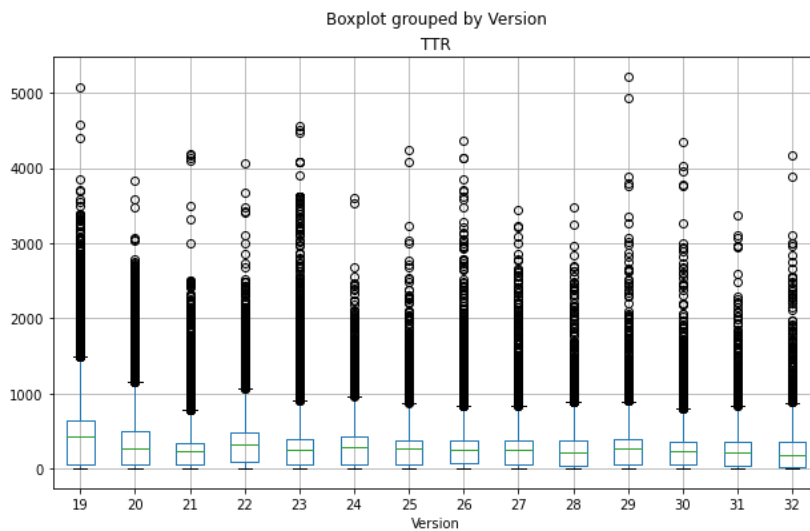


@FunnelFiasco

CC BY-SA 4.0

On a log scale, the graph looks pretty linear.
That's neat.

TTR BY RELEASE

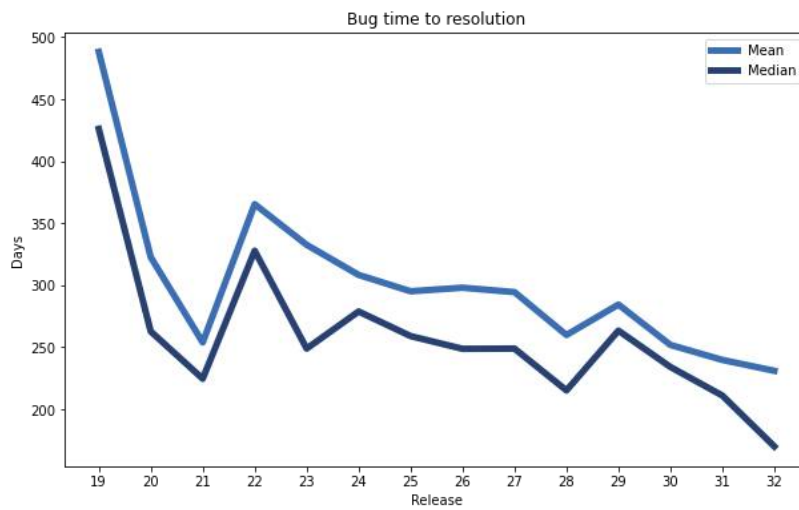


@FunnelFiasco

CC BY-SA 4.0

Looking at TTR by release, it's clear that there are a lot of high outliers. But we seem to be closing bugs faster over time.

MEAN/MEDIAN TTR



@FunnelFiasco

CC BY-SA 4.0

Looking at the mean and median, this is clearly the case.

10 LOWEST MEDIAN TTR BY COMPONENT

Component	TTR (days)
golang-github-coreos-systemd	0.003
kipi-plugins	0.003
wmweather	0.003
darkice	0.004
marble-widget	0.005
php-simplesamlphp-saml2_1	0.006
deepin-tool-kit	0.007
gwe	0.008
mingw-crt	0.009
ipw2200-firmware	0.010

@FunnelFiasco

CC BY-SA 4.0



Some components are lightning fast. Probably because the bugs are opened by a maintainer who immediately resolves it.

10 HIGHEST MEDIAN TTR BY COMPONENT

Component	TTR (days)
esound	3610.7
gtk-sharp2	3610.7
java_cup	3610.7
mx	3610.7
fonts-KOI8-R	3115.6
libdaemon	3107.9
xorg-x11-xfs	3101.7
libgnomeui	2948.3
fedora-package-config-smart	2742.0
libzdb	2724.7

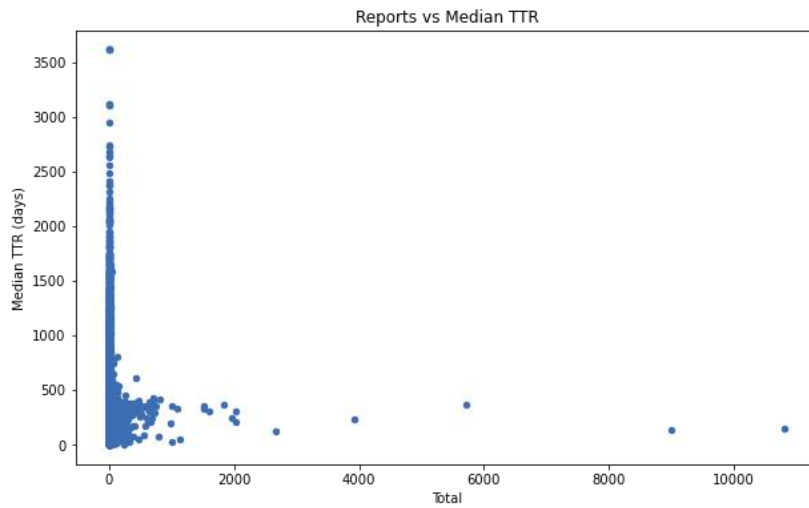
@FunnelFiasco

CC BY-SA 4.0



Some components have a 10-year median closure time. That's...a lot.

REPORTS VS MEDIAN TTR



@FunnelFiasco

CC BY-SA 4.0

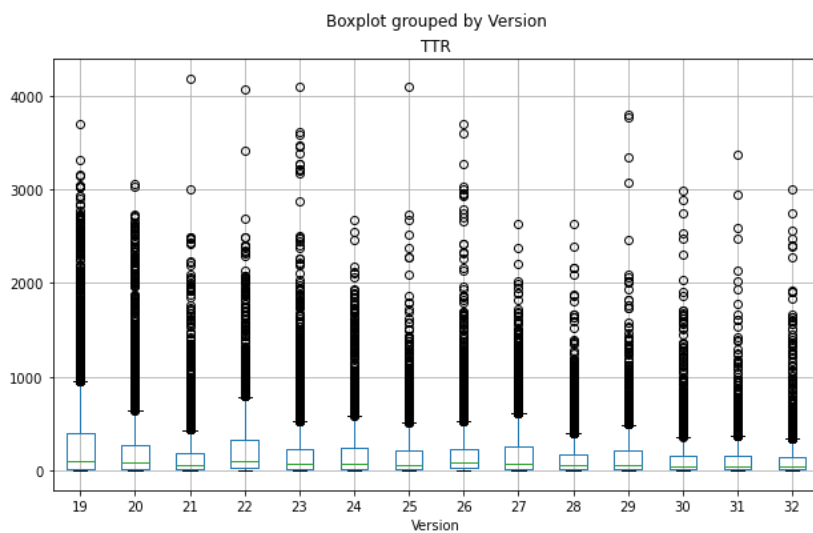
Does the TTR vary by the number of bug reports?
Not really.

HAPPY TTR

CC BY-SA 4.0

Let's do the same thing, but just for the happy closures.

HAPPY TTR BY RELEASE

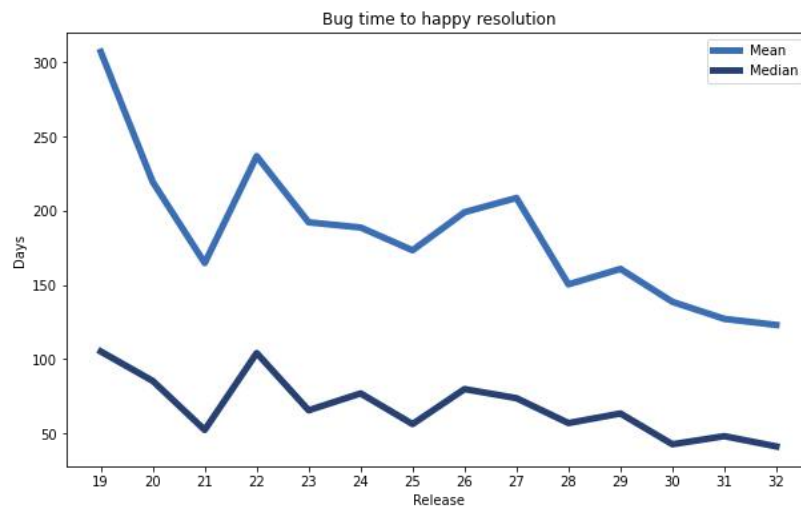


@FunnelFiasco

CC BY-SA 4.0



MEAN/MEDIAN HAPPY TTR BY RELEASE



@FunnelFiasco

CC BY-SA 4.0

We're improving at fixing bugs!

10 LOWEST MEDIAN HAPPY TTR

Component	TTR (days)
biosdevname	0.001
golang-github-coreos-systemd	0.003
kde-cli-tools	0.005
spice-html5	0.005
kf5-kirigami	0.007
gwe	0.008
php-tcpdf	0.008
date	0.011
system-config-audit	0.013
akregator	0.014

@FunnelFiasco

CC BY-SA 4.0



10 HIGHEST MEDIAN HAPPY TTR

Component	TTR (days)
module-init-tools	3694.8
gamin	3198.8
bowtie	3002.8
nodejs-request	2718.8
radeon-tool	2718.5
poster	2684.5
tidy	2466.2
nodejs-jade	2410.1
nodejs-highlight-js	2389.0
nodejs-qs	2233.9

@FunnelFiasco

CC BY-SA 4.0

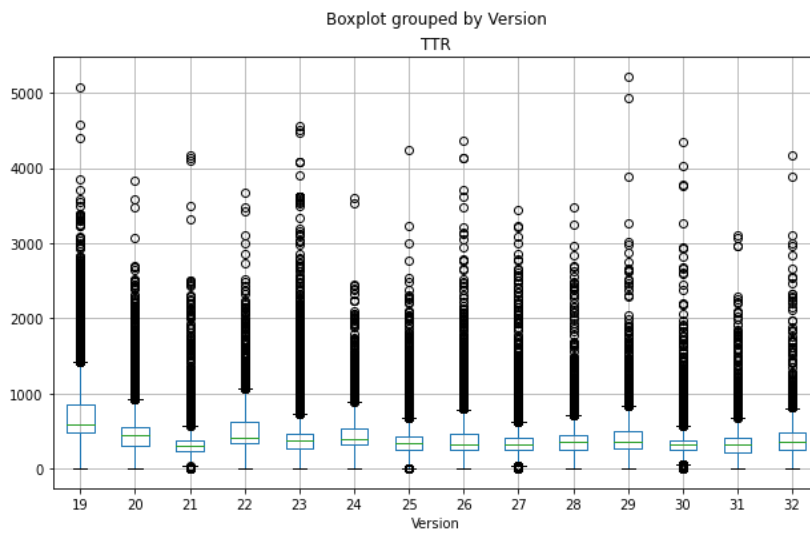


You might have to wait years, but we can still get bugs fixed.

SAD USER TTR

CC BY-SA 4.0

SAD USER TTR BY RELEASE

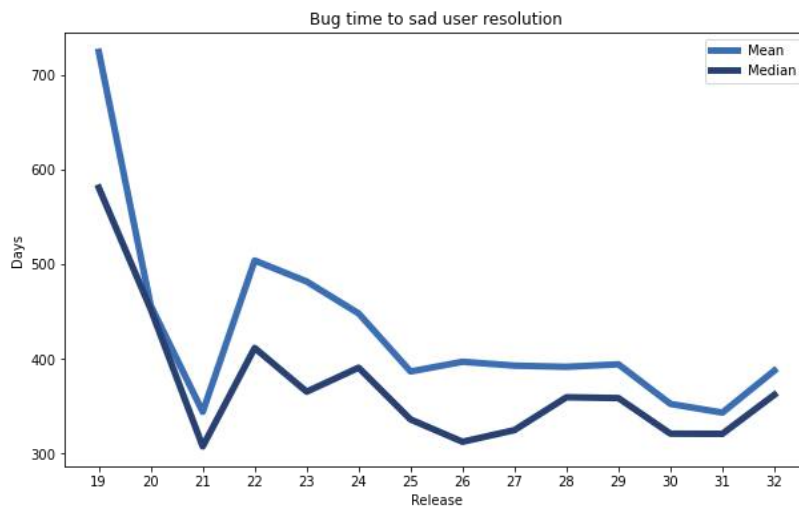


@FunnelFiasco

CC BY-SA 4.0



ME{AN,EDIAN} SAD USER TTR BY RELEASE



@FunnelFiasco

CC BY-SA 4.0

The large drop in sad user TTR from F19 to F20 is probably due to a large backlog of EOL closures.

10 LOWEST SAD USER TTR

Component	TTR (days)
libmpd	0.002
darkice	0.004
relval	0.006
apache-commons-compress	0.008
perl-local-lib	0.009
polkit-qt	0.013
glassfish-hk2	0.017
xmlunit	0.017
jackrabbit	0.020
ssldump	0.023

@FunnelFiasco

CC BY-SA 4.0



10 HIGHEST SAD USER TTR

Component	TTR (days)
perl-RPM2	4128.4
dmidecode	3610.7
epic	3610.7
esound	3610.7
gtk-sharp2	3610.7
java_cup	3610.7
libtheora	3610.7
mx	3610.7
Pyrex	3610.6
ytnef	3481.7

@FunnelFiasco

CC BY-SA 4.0



WHAT'S NEXT?

CC BY-SA 4.0

WHY DIDN'T YOU INCLUDE ____?

- Version changes
- Assignees
- Freeze process

@FunnelFiasco

CC BY-SA 4.0



Bugzilla queries don't report a count of version changes, but that would be very interesting to know.

I didn't include assignees because I didn't want a bunch of email addresses in the file.

I could include a count of freeze process related bugs, I just didn't do that in this query.

COMING SOON

- Community Blog post(s)
- Your theories
- Probably more graphs!

@FunnelFiasco

CC BY-SA 4.0



EXPLORE IT YOURSELF

- <https://pagure.io/fedora-pgm/fedora-bug-data>



@FunnelFiasco

CC BY-SA 4.0

